Untitled

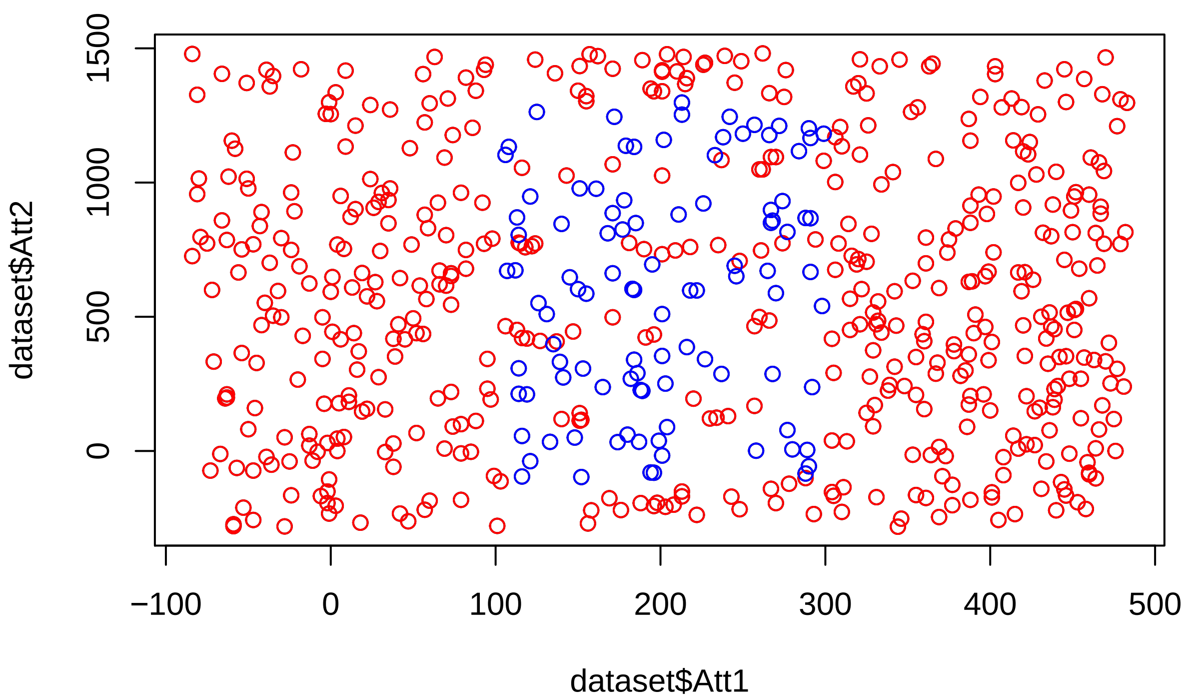
Análisis del efecto del desbalanceo en problemas de clasificación

En primer lugar trabajaremos con los datos subclus.txt

```
#load dataset subclus
dataset <- read.table("subclus.txt", sep=",")
#dataset <- read.table("circle.txt", sep=",")
colnames(dataset) <- c("Att1", "Att2", "Class")
summary(dataset)
```

```
##       Att1             Att2            Class
##  Min.   :-84.00   Min.   :-282.0   negative:500
##  1st Qu.: 65.75   1st Qu.: 155.8   positive:100
##  Median :213.00   Median : 572.5
##  Mean   :214.06   Mean   : 574.5
##  3rd Qu.:365.50   3rd Qu.: 961.2
##  Max.   :483.00   Max.   :1481.0
```

```
# visualize the data distribution
plot(dataset$Att1, dataset$Att2)
points(dataset[dataset$Class=="negative",1],dataset[dataset$Class=="negative",2],col="red")
points(dataset[dataset$Class=="positive",1],dataset[dataset$Class=="positive",2],col="blue")
```



Podemos observar como el imbalance ratio tiene un valo de 0.2 que tay y como se observa en el plot nateiror representa que nos encontramos delante de un dataset desbalanceado.

```
imbalanceRatio(dataset) #
```

```
## [1] 0.2
```

A continuación creremos las particiones de training y test del datatset subclus para poder aplicar distintos algoritmos de undersampling y oversampling para una comparación posterior.

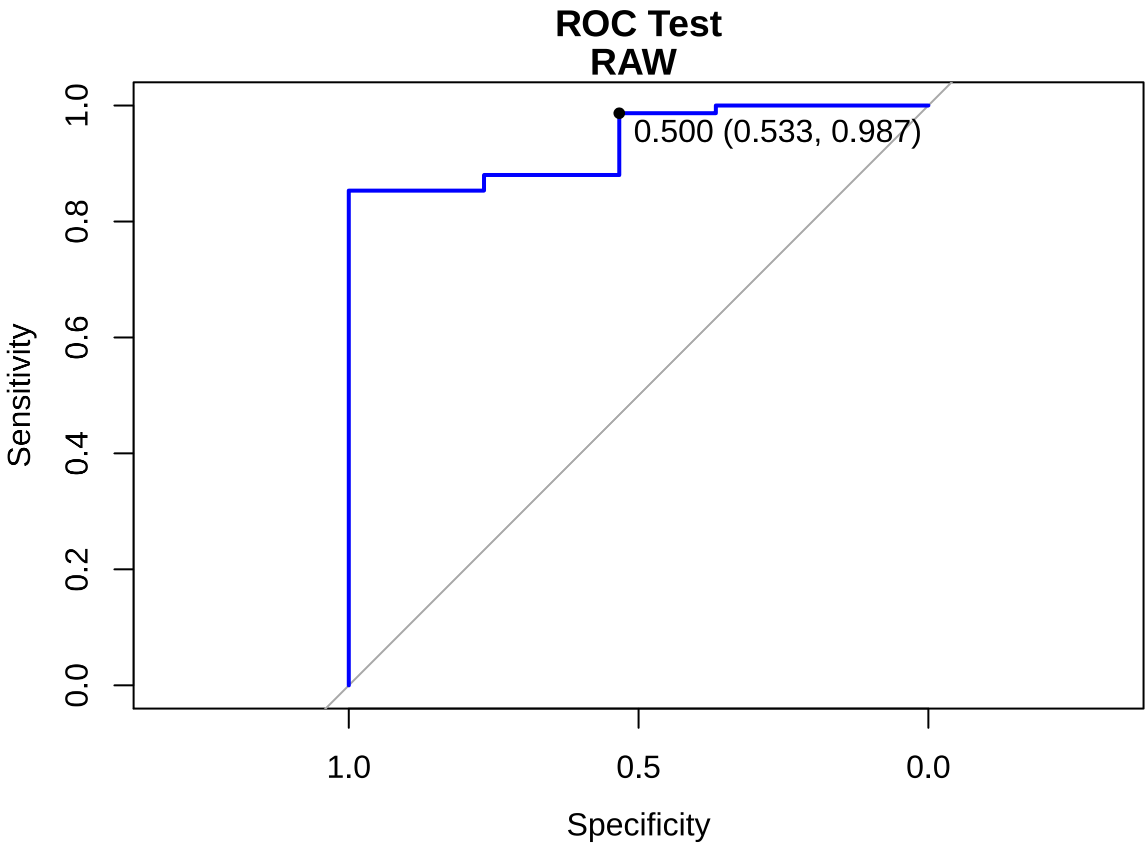
```r
#Create Data Partition
set.seed(42)
dataset$Class <- relevel(dataset$Class,"positive")
index <- createDataPartition(dataset$Class, p = 0.7, list = FALSE)
train_data <- dataset[index, ]
test_data  <- dataset[-index, ]

#Execute model ("raw" data)
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction = twoClassSummary)

model.subclus.raw <- learn_model(train_data,ctrl,"RAW ")

## Aplicamos el modelo con Random Undersampling
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction = twoClassSummary, sampling = "down") #RUS
model.subclus.us <- learn_model(train_data,ctrl,"US ")

## Aplicamos el modelo con Random Oversampling
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction = twoClassSummary, sampling = "up") #ROS
model.subclus.os <- learn_model(train_data,ctrl,"OS ")

## Aplicamos el modelo con Synthetic Minority Oversampling Technique
ctrl <- trainControl(method="repeatedcv",number=5,repeats = 3,
                     classProbs=TRUE,summaryFunction = twoClassSummary, sampling = "smote") #SMOTE
model.subclus.smt <- learn_model(train_data,ctrl,"SMT ")
```

```
## Loading required package: grid
```

```r
cm.subclus.raw <- test_model(test_data,model.subclus.raw,"RAW ")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive       16        2
##   negative       14      148
##
##                Accuracy : 0.9111
##                  95% CI : (0.8597, 0.9483)
##     No Information Rate : 0.8333
##     P-Value [Acc > NIR] : 0.001979
##
##                   Kappa : 0.619
##  Mcnemar's Test P-Value : 0.005960
##
##             Sensitivity : 0.53333
##             Specificity : 0.98667
##          Pos Pred Value : 0.88889
##          Neg Pred Value : 0.91358
```

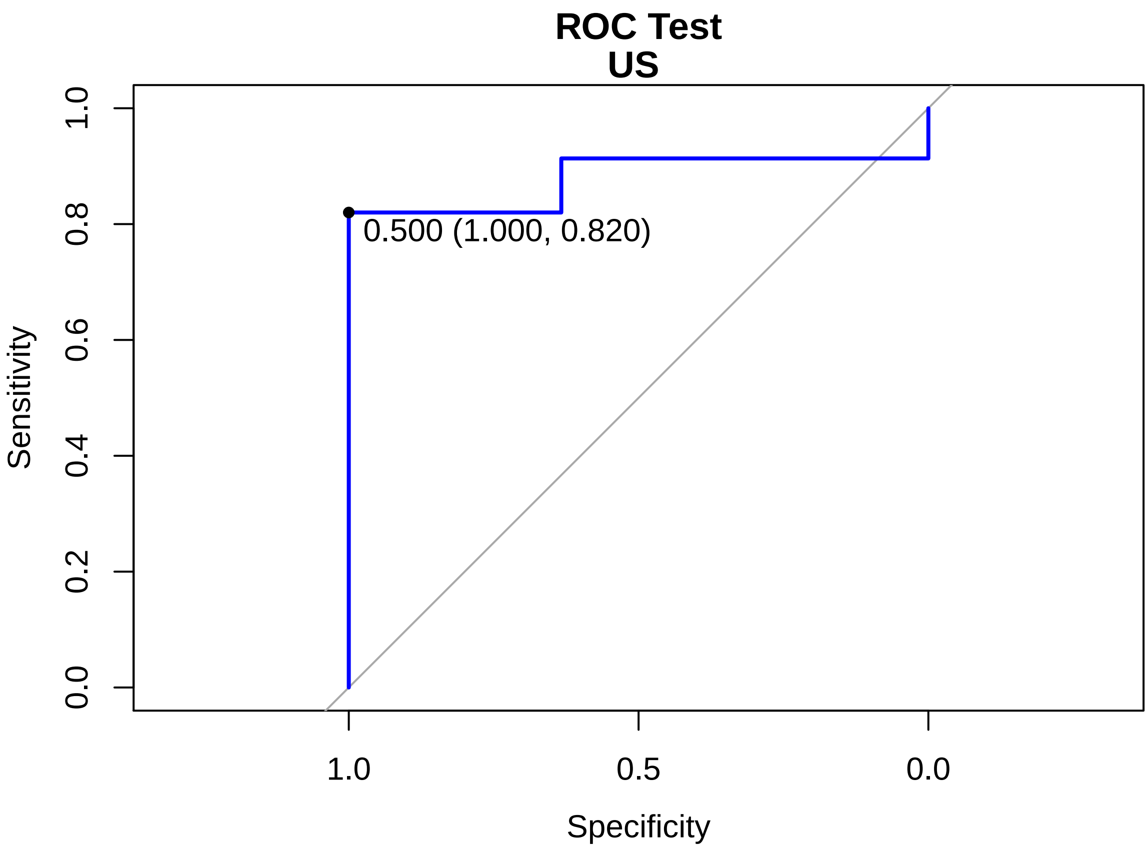
```
##               Prevalence : 0.16667
##           Detection Rate : 0.08889
##     Detection Prevalence : 0.10000
##        Balanced Accuracy : 0.76000
##
##         'Positive' Class : positive
##
```




```r
cm.subclus.us <- test_model(test_data,model.subclus.us,"US ") #undersampling
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive       30       27
##   negative        0      123
##
##                Accuracy : 0.85
##                  95% CI : (0.7893, 0.8988)
##     No Information Rate : 0.8333
##     P-Value [Acc > NIR] : 0.3146
##
##                   Kappa : 0.6029
##  Mcnemar's Test P-Value : 5.624e-07
##
##             Sensitivity : 1.0000
##             Specificity : 0.8200
##          Pos Pred Value : 0.5263
```

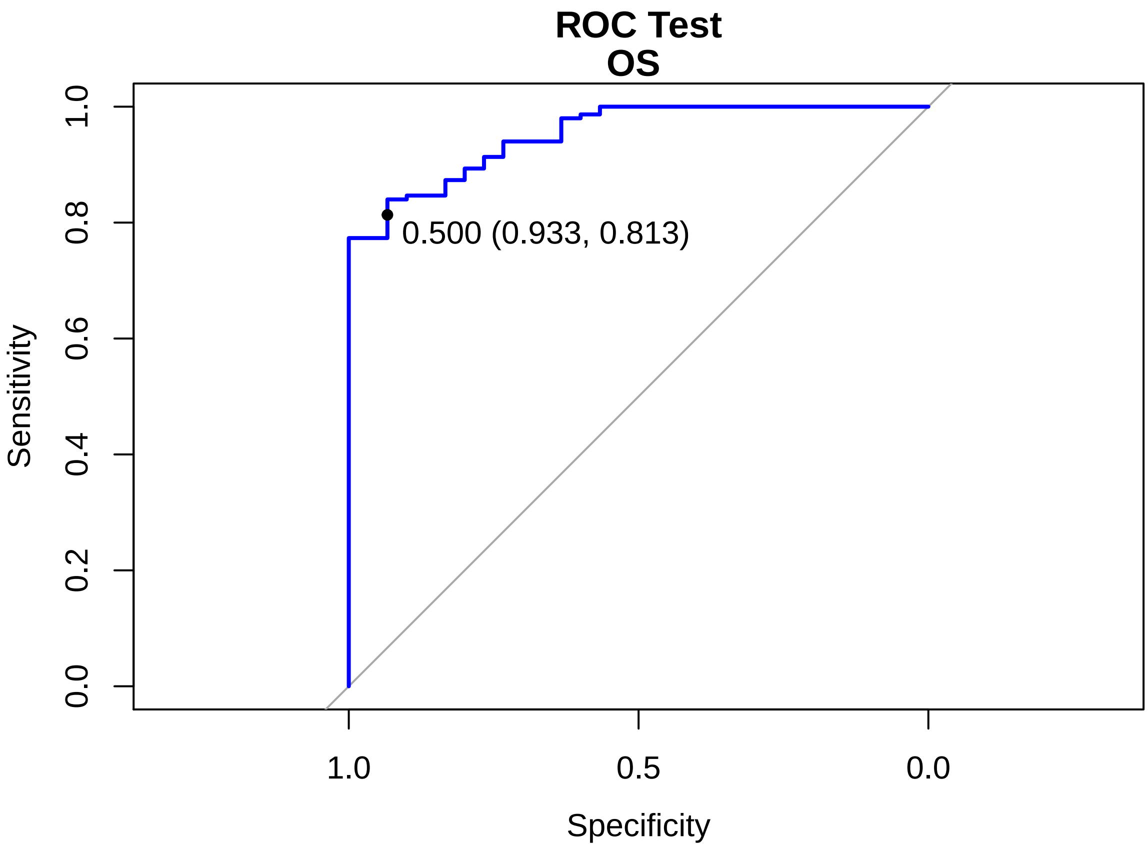
```
##          Neg Pred Value : 1.0000
##              Prevalence : 0.1667
##          Detection Rate : 0.1667
##    Detection Prevalence : 0.3167
##       Balanced Accuracy : 0.9100
##
##        'Positive' Class : positive
##
```




```r
cm.subclus.os <- test_model(test_data,model.subclus.os,"OS ") #oversampling
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive       28       30
##   negative        2      120
##
##                Accuracy : 0.8222
##                  95% CI : (0.7584, 0.8751)
##     No Information Rate : 0.8333
##     P-Value [Acc > NIR] : 0.6972
##
##                   Kappa : 0.534
##  Mcnemar's Test P-Value : 1.815e-06
##
##             Sensitivity : 0.9333
##             Specificity : 0.8000
```

```
##          Pos Pred Value : 0.4828
##          Neg Pred Value : 0.9836
##              Prevalence : 0.1667
##          Detection Rate : 0.1556
##    Detection Prevalence : 0.3222
##       Balanced Accuracy : 0.8667
##
##        'Positive' Class : positive
##
```




```
cm.subclus.smt <- test_model(test_data,model.subclus.smt,"SMT ")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction positive negative
##   positive       26       21
##   negative        4      129
##
##                Accuracy : 0.8611
##                  95% CI : (0.8018, 0.9081)
##     No Information Rate : 0.8333
##     P-Value [Acc > NIR] : 0.185104
##
##                   Kappa : 0.5924
##  Mcnemar's Test P-Value : 0.001374
##
##             Sensitivity : 0.8667
```

```
##               Specificity : 0.8600
##            Pos Pred Value : 0.5532
##            Neg Pred Value : 0.9699
##                Prevalence : 0.1667
##            Detection Rate : 0.1444
##      Detection Prevalence : 0.2611
##         Balanced Accuracy : 0.8633
##
##          'Positive' Class : positive
##
```